REMARKS

I. Introduction

With the addition of claim 34, claims 1 to 34 are now pending.

Applicants thank the Examiner for indicating that the amendments to the drawings and title successfully overcame the objections.

II. Oath/Declaration

With respect to paragraph three (3) of the Office Action, the Declaration was asserted to be defective for lacking a mailing address for each inventor, and for including a non-initialed alteration. In regards to the mailing address, it is respectfully submitted that the Declaration includes the mailing address of each inventor since the addresses listed in the Declaration are address at which the inventors customarily receive their mail. In regards to the non-initialed alteration, as Michael P. Paul (Reg. No. 53,443) discussed with Examiner Eddie Chan on April 11, 2005, a copy of the initialed Declaration is forthcoming. The Examiner agreed that this response would be responsive.

III. Rejection of claim 32 under 35 U.S.C. § 112

Claim 32 stands rejected under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. In this regard, claim 32 has been amended to better clarify the claimed subject matter, and is believed to be self-explanatory. It is respectfully submitted that claim 32 as presented is clear and definite. According, withdrawal of the rejection of claim 32 under 35 U.S.C. § 112 is respectfully requested.

IV. Rejection of claims 1 to 33 under 35 U.S.C. § 102(b)

Claims 1 to 33 stand rejected under 35 U.S.C. § 102(b) as anticipated by Vajapeyam et al., "Improving Superscalar Instruction Dispatch and Issue by Exploiting Dynamic Code Sequences," Computer Architecture 1997 ("Vajapeyam"). It is respectfully submitted that claims 1 to 33 are not anticipated by Vajapeyam for at least the following reasons.

Claim 1 as presented relates to a method for renaming a source for use with a processor, which includes providing at least one instruction, building **instruction dependency chain information** based on the at least one instruction, caching the at least one

instruction with only the **instruction dependency chain information** to provide cached instruction information, renaming a register based on the cached instruction information to provide a renamed register, and multiplexing the instruction dependency chain information and the renamed register to rename the source. In this regard, the present application provides, for example, the following:

> [T]he **dependency information may be cached** to eliminate the use of the prioritized content-addressable-memories (CAMs) with the renamer arrangement and/or method of Figure 2. Caching the **dependency chain information** is believed to be effective for providing sequential allocation, and may be done using the dependency information field arrangement 400 of Figure 4. In the field arrangement 400, the first source Src1 and the second source Src2 information are provided using three (3) bits and the destination field Dst is provided using one (1) bit, which collectively provides the cached dependency chain information for use in the fast renaming apparatus, method and system of Figures 5 and 6.
>
> In particular, in the pre-decode information for providing fast renaming, the first source Src1 bits and the second source Src2 bits **indicate upon which instruction in the rename window the particular instruction depends.** A special encoding (such as, for example, '111) may indicate or otherwise denote that the specific source Src is not produced by any previous instruction in the rename window. The destination Dst bit indicates that this instruction updates the register alias table (RAT). Using this pre-decode information, the content-addressable-memories (CAMs) may be replaced by an "adder" or concatenating arrangement as shown in Figure 5. The fast renaming algorithm depends on sequential ID allocation for the renamed instructions. In particular, the system may determine a final virtual ID by concatenating the upper bits from the renamer with the lower bits from the instruction cache. It is believed that the latter approach may be faster and/or easier to implement, but may require allocation of virtual registers in larger, predefined groups having a size that depends on a power of 2.

(Specification, page 13, line 11 to page 14, line 2) (Emphasis added). Hence, instruction dependency information indicates which other instruction a particular instruction depends upon and where that other instruction is stored, rather than the actual renaming of a register.

The Vajapeyam reference, by contrast, relates to improving superscalar instruction dispatch by partitioning an instruction window into multiple blocks, partitioning a register file into a global file and several local files, and recording **register renaming**

information in a trace cache. (See Vajapeyam, page 1, columns 1 to 2). In this regard, Figure 5 of Vajapeyam shows a typical entry in the trace cache, in which information is recorded that indicates which "live-on-entry" and "live-on-exit" registers are to be renamed before their use, but does not include **chain** information that indicates upon which other **instruction** the particular instruction depends. (See Vajapeyam, page 5, columns 1 and 2, Figure 5 and related text). Hence, Vajapeyam deals with a "block renaming" scheme, which requires both local and global renaming tables. The present invention, by contrast, merely requires a single MAP table and an instruction window because only the dependency information is stored with the trace line (i.e., dependency chain information) and thus the dependency detection process (which is slow and serial) can be separated from temporary space allocation (which may be performed in parallel).

Such differences between Vajapeyam and the subject matter of claim 1 are exemplified by Vajapeyam's lack of a solution to the "clobber" problem, in which during a re-execution attempt a mapped physical register becomes either not associated with the original mapping, or mapped to a different logical register. In this regard, the actual mapping of logical to physical register depends on the machine state, for example, on the previously executed instructions and their latency, outside the cache line. The subject matter of claim 1 overcomes the clobber problem since only dependency information is stored inside the cache line (the dependency information is a pure function of the instruction stream), and an efficient process is provided to convert the dependency chain into the actual register mapping. Thus, rather than sending source S2 to execution with its mapping "as is" as proposed in Figure 5 of Vajapeyam, the source S2 may be first concatenated with the value of the incrementer to form the full virtual-physical register identifier, and later at the execution stage, the source S2 may receive the actual physical mapping. Hence, the clobbering problem can be resolved if the actual mapping of logical to virtual-physical register is performed during the renamer stage, which is consistent with the method for renaming as described by claim 1, but not with what is discussed in Vajapeyam.

It is therefore respectfully submitted that Vajapeyam does not in anyway disclose, or suggest, caching at least one instruction only with instruction dependency chain information, as recited in claim 1. Accordingly, Vajapeyam fails to identically disclose all of the features of claim 1 — as it must to support an anticipation rejection. It is therefore respectfully submitted that claim 1 is allowable for at least these reasons.

Claims 2 to 7 and 29 to 33 depend either directly or indirectly from claim 1, and are therefore allowable for at least the same reasons as claim 1.

Claims 8, 15, and 22 recite features analogous to claim 1, and are therefore allowable for essentially the same reasons as claim 1.

Claims 9 to 14 depend either directly or indirectly from claim 8, and are therefore allowable for at least the same reasons as claim 8.

Claims 16 to 21 depend either directly or indirectly from claim 15, and are therefore allowable for at least the same reasons at claim 15.

Claims 23 to 28 depend either directly or indirectly from claim 22, and are therefore allowable for at least the same reasons that claim 22 is allowable.

New claim 34 does not add any new matter and is supported by the present application, including the specification. Claim 34 depends directly from claim 1, and is therefore allowable for at least the same reasons that claim 1 is allowable.

In sum, it is respectfully submitted that claims 1 to 34 are allowable for at least the reasons discussed above. Withdrawal of the anticipation rejections of claims 1 to 34 is therefore respectfully requested.

CONCLUSION

In view of the foregoing, it is respectfully submitted that all of the presently pending claims are allowable. It is therefore respectfully requested that the objections and rejections be withdrawn since they have been obviated. All issues raised by the Examiner having been addressed, an early and favorable action on the merits is respectfully requested.

Respectfully submitted,
KENYON & KENYON

Dated: 4/11/2005 By: _____

Aaron C. Deditch
Reg. No. 33,865

One Broadway
New York, NY 10004
(212) 425-7200
CUSTOMER NO. 26646